Hose 解题报告

【问题简述】

给定 n 个数{A_1,A_2……A_N}，m 个询问(L,R)。
对于每个询问，请你计算

$$P = 从A_L, A_{L+1} \dots A_R 中随机取出两个数相同的概率$$

数据范围：$1 \le n, m, A_i \le 50000$。

【问题的转化】

令 T_i 表示数 i 在$A_L, A_{L+1} \dots A_R$中的出现次数，容易看出：

$$P = \frac{\sum C_{T_i}^2}{C_{R-L+1}^2} = \frac{\sum T_i * (T_i - 1)/2}{(R-L+1)(R-L)/2} = \frac{\sum {T_i}^2 - \sum T_i}{(R-L+1)(R-L)}$$

由于$\sum T_i = R - L + 1$，故问题等价于对于每个询问求$\sum {T_i}^2$。

后文的讨论均针对转化后的问题，这里先做几个便于叙述的约定：

1. 区间[L, R]表示一个询问(L, R)。
2. 求解区间[L, R]表示求区间[L, R]的答案。
3. 区间[L, R]的答案表示$\sum {T_i}^2$ (T_i为 i 在[L, R]出现的次数)。
4. i 在[L, R]内的出现次数表示数 i 在$A_L, A_{L+1} \dots A_R$中的出现次数。

【朴素算法】

对于每个区间，循环统计该区间内每个数的出现次数并计算答案，时间复杂度为 O($\sum R - L + 1$) ≅O(nm)，较好的实现可以得到 30 分。

【初步分析】

朴素算法的效率之所以低下在于没有充分利用询问区间的重合性，这引导我们思考求解区间问题的利器——线段树。

使用线段树的关键条件是区间信息的可合并性，即对于两个相邻的区间[a,b]与[b+1,c]，要能方便的由它们的信息计算出区间[a,c]的信息。

考虑本题，要记录哪些信息才能方便的合并呢？

区间的答案？

设数 i 在[a,b]中的出现了 x 次，在[b+1,c]中出现了 y 次，由于$x^2 + y^2 \neq (x+y)^2$，故无法从[a,b]的答案与[b+1,c]的答案求出[a,c]的答案。

每个数在区间内的出现次数？

由于最多有 n 个不同的数，所需信息太多。

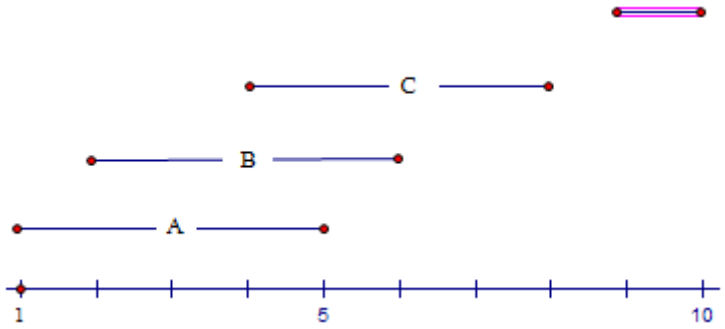
平方运算的存在成了线段树无法打破的坚冰，思路似乎进入了死胡同。

【另辟蹊径】

回忆曾做过的一道求区间 K 大数的题目[1]：

> 给定 n 个互不相等的数 $P_1,P_2,P_3\cdots\cdots P_n$ 以及 m 个询问(L,R,K)，求 $P_L,P_{L+1}\cdots\cdots P_R$ 中第 K 小的数。（保证所有(L,R)互不包含）

由于区间互不包含，故若将区间按左端点排序，则右端点递增。



如上图所示，n=10，m=4，4 个询问分别是A(1,5) B(2,6) C(4,8) D(9,10)[2]。

假设我们已有一棵平衡树 T 保存了区间 A（P_1 至 P_5），用平衡树的查找操作容易找到其中的 K 小数。

可以发现，区间 A 与区间 B 有很大的重合部分，如果在 T 中插入 P_6，删除 P_1，则 T 将保存区间 B（P_2 至 P_6），求出其 K 小数。类似的，插入 P_7 与 P_8，删除 P_2 与 P_3，则 T 将保存区间 C。C 与 D 尽管没有重合部分，仍可以类似地处理，插入 P_9 与 P_{10}，删除 P_4 至 P_8，则 T 将保存区间 D。

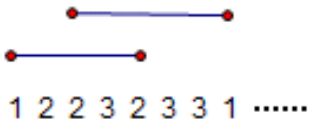
这样从左至右不停的插入删除，使得 T 依次保存每个区间，并求出询问的 K 小数便可解决本题。

由于每个数 P_i 至多在平衡树 T 中插入与删除一次，而每个询问只需要一次查询操作，故总的时间复杂度为$O((n+m)\log n)$。

【核心操作】

回到本题中，假设区间互不包含，则很容易设计出线性算法：

将区间按左端点排序，从左至右扫描，任意时刻保存当前区间每个数的出现次数 T_i 以及$S=\sum T_i^2$。每次从一个区间$[a,b]$转移到另一个区间$[c,d]$时，插入$(b,d]$内的数，删除区间$[a,c)$内的数，并维护 T_i 以及 S。



以上图数据为例，一行数为序列 A，询问为(1,5)与(3,8)。设已求出询问(1,5)内的信息：$T_1=1\ T_2=3\ T_3=1\ S=11$。要从(1,5)转到(3,8)，需要插入 A_6,A_7 与 A_8，删除 A_1 与 A_2，具体的操作过程如下：

---

[1] 野生动物园，vijos1081

[2] 由于具体的 K 对算法没有影响，略去，后面直接用 K 表示

插入 A_6=3：T_3 由 1 变为 2，S 变为$11-1^2+2^2=14$。
插入 A_7=3：T_3 由 2 变为 3，S 变为$14-2^2+3^2=19$。
插入 A_8=1：T_1 由 1 变为 2，S 变为$19-1^2+2^2=22$。
删除 A_1=1：T_1 由 2 变为 1，S 变为$22-2^2+1^2=19$。
删除 A_2=2：T_2 由 3 变为 2，S 变为$19-3^2+2^2=14$。
这样便得到了区间(3,8)的信息：$T_1=1\ T_2=2\ T_3=3\ S=14$。

类似的，由于每个数最多插入或删除一次，而每次插入或删除可以在 O(1)的时间内完成，故总的时间复杂度为$O(n+m)$。

尽管该算法不能直接应用于本题，但 O(1)的维护操作着实是个好消息，后文的两种算法均以该操作为基础，故请读者务必熟悉。

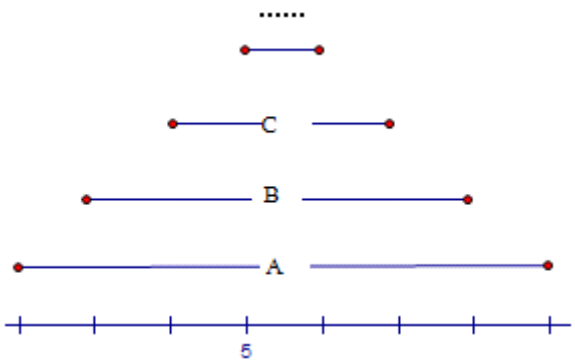
【算法一：序列划分】

采用化整为零的思想，将区间集合 Q 分为若干子集 Q_1,Q_2……Q_k 使得：

$$\forall i\in[1,k]，Q_i\text{内的区间互不包含，且}\bigcup_{i=1}^{k}Q_i=Q$$

显然，每个 Q_i 内的区间可以在 O(n)的时间内求解，故可以在 O(nk)的时间内求解所有的区间，如果 k 较小的话，这将是一个优秀的算法。

但容易发现，很多情况下 k 并不小，最极端的情况如下：



任两个区间有包含关系，无论如何只能分为 m 个子集，那么该算法便与朴素算法无异了，时间复杂度退化到 O(nm)。

实际上该例子很容易解决，先插入区间 A 中所有数以得到区间 A 的信息，然后删除 A_2 与 A_9 便可得到区间 B 的信息，再删除 A_3 与 A_8 便可得到区间 C 的信息。依此类推，由于每个数至多插入和删除一次，故若给定的区间两两有包含关系则仍可以在 O(n)时间内解决。

比较这两种情况，共同点是左端点不减，区别是右端点的单调性，前者是不降，后者是不升，这恰恰对应了动态规划问题中的不降子序列与不升子序列！

具体来说，将区间以左端点为第一关键字，右端点为第二关键字排序，右端点作为其关键字，得到一个长 m 的序列 Z。

则 Z 的任一个不降子序列对应的区间集合满足任两个区间要么端点重合要么互不包含，而 Z 的任一个不升子序列对应的区间满足任两个区间有包含关系，显然这两种区间集合都可以在 O(n)的时间内求解。

故问题的关键在于将序列 Z 划分为最上尽量少的不降子序列或不升子序列，这个问题并不是那么容易解决的[3]，考虑降低目标，求一个较优解。

下面介绍一种基于贪心的算法：

```
While  Z 不为空  do
    求 Z 的最长不降子序列 X。
    求 Z 的最长不升子序列 Y。
    令 Max = X 与 Y 中长度较大的。
    求解 Max 对应的区间集合。
    从 Z 中删除 Max。
```

该算法的时间复杂度取决于划分出的序列个数（即 While 循环进行的次数），若划分出 K 个序列，则该算法的时间复杂度为 O(K(mlogm+n))[4]。

实际上 K 的上限为 $O(\sqrt{m})$（证明见后），且该上限很容易达到，故序列划分算法可以在 $O(\sqrt{m}(mlogm+n))$的时间内解决本题，期望得分 60+。

【K 上限的证明】

dilworth 定理[5]：$<X,\leq>$是偏序集，r 是最长反链的长度，则 X 可以划分成 r 个但不能再少的链。

对应到序列 P 中，一条链即一个上升子序列，一条反链即一个下降子序列。令 x 为最长上升子序列的长度，y 为最长下降子序列的长度。则 P 可以划分成 y 个长度不超过 x 的上升子序列，有$y * x \geq m$，进而有$\max\{x,y\} \geq \sqrt{m}$，令 T(m)为长 m 的序列所划出的子序列个数，有：

$$T(m) \leq T\left(m-\sqrt{m}\right)+1 \quad T(0)=0$$

解递归方程得：$T(m) \leq 2\sqrt{m}$，即 K 的上限为 $O(\sqrt{m})$，命题得证。

【算法二：类 TSP[6]】

算法一主要的时间耗费在寻找最长单调序列上，但这并不是本质需要，事实上我们的目的在于使得维护操作的总次数尽量少。

考虑两个任意位置的区间$[L_1,R_1]$与$[L_2,R_2]$，已经维护好了前者的信息，要得到后者的信息，只需要插入或删除$[L_1,L_2)$与$(R_1,R_2]$中的数。

具体来说，若$L_1 \leq L_2$则删除$[L_1,L_2)$内的数，否则插入$[L_1,L_2)$内的数。若$R_1 \leq R_2$则插入$(R_1,R_2]$内的数，否则删除$(R_1,R_2]$内的数。无论是哪种情况，所需的操作数总是$|L_1-R_1|+|L_2-R_2|$。

利用数形结合的思想，将区间(L,R)看做平面上的一个点(x=L,y=R)，那么在两个区间之间转化所需的操作数等于其对应点间的 Manhattan 距离，求解所有询问的一个顺序等价于从点(0,-1)出发经过所有点的一条哈密尔顿路，总操作数则等价于该路径的长度！

故问题的关键在于求一条长度最小的哈密尔顿路，即经典的 TSP 问题（当然，

---

[3] 似乎是 NP-hard 的

[4] 最长**子序列利用动态规划+树状数组解决，时间复杂度 mlogm

[5] 该定理的介绍可以在大多数组合数学书中找到

[6] Travelling salesman problem，又称货郎担问题

是 NP-hard 的）。与算法一类似，我们不得不降低条件，求较优解。

提到 TSP 问题，首先想到的自然是模拟退火，遗传算法等经典非完美算法，尽管它们求出的解很优，但无论是时间复杂度还是编程复杂度都不适用本题。

分析本问题的特点，费用函数 Dis(i,j)为 i 与 j 的 Manhattan 距离，满足三角不等式$\mathrm{Dis}(i,j)+\mathrm{Dis}(j,k)\geq\mathrm{Dis}(i,k)\ \ i,j,k\in V$。这里介绍一种利用该特性的近似算法，使得对于任何数据求出的解均不超过最优解的两倍（证明见后）。

1. 将区间集转为点集 V，加入点 O(0,-1)。
2. 对于任意两点 i 与 j 连一条边，长度为其 Manhattan 距离。
3. 求出该图 G 的最小生成树 T。
4. 以 O 为根先序遍历树 T，得 DFS 序列 H。
5. 以 H 为所求哈密尔顿路，即按照 H 的顺序求解所有区间。

由于 H 的长度不超过最优解的两倍，而序列划分算法等价于构造了一条长度不超过 $O(n\sqrt{m})$的路径，故 H 的长度上限为 $O(n\sqrt{m})$。又 Manhattan 距离最小生成树可以用区域分类法[7]在 O(mlogm)的时间内求出，故类 TSP 算法可以在 $O(m\log m+n\sqrt{m})$的时间内解决本题，期望得分 100。

【解不超过最优解两倍的证明[8]】

设最短哈密尔顿路为 H*，其长度为 c(H*)，H 的长度为 c(H)，T 的边权和为 c(T)。由于 H*也是 G 的一个生成树，故有 c(H*)≥c(T)。

对 T 所做的完全遍历指访问 T 的顶点 i 时记录点 i 一次，而在结束 i 的一棵子树的访问并沿途返回时也记录点 i。设 W 为对 T 进行完全遍历得到的序列，由于 W 经过 T 的每条边恰两次，故 c(W)=2c(T)<=2c(H*)。

然而 W 还不是一个哈密尔顿路，它访问了 G 中某些顶点多次。由于费用满足三角不等式，可以在 W 的基础上删去已访问的点，而不会使得 c(W)增加。

具体来说，删除点 u 和 v 间的点 w 时，用边(u,v)替换 u 到 v 的路，反复这样删去 W 中多次访问的顶点即可得到 T，且 c(T)<=c(W)<=2c(H*)，命题得证。

【总结】

通过本题，我们发现了这类只有询问没有修改的统计型问题的新思路，即通过特殊顺序求解询问，使得相邻求解的询问重合程度尽量高。

为了得到一个优秀的求解顺序，提出了序列划分与类 TSP 两种算法。两者均不拘泥于求最优解，达到了时间复杂度与解的质量的均衡。而两者相比，前者与树的路径剖分相似，利用了偏序集的一些性质；后者利用数形结合更加精准的描述了问题模型，并结合 TSP 问题的近似算法得到了非常优秀的结果。

只要我们充分挖掘，再平淡无奇的题目中也会绽放思维的火花。

[7] 见附录 1

[8] 引自《算法设计与分析》P331

【附录】

1. 平面点曼哈顿距离最小生成树的 nlogn 算法： 
1.rar

2. 算法二的参考程序(有注释)： 
Hose.cpp

【参考文献】

1. 《算法艺术与信息学竞赛》 刘汝佳，黄亮
2. 《Efficient minimum spanning tree construction without Delaunay triangulation》 Hai Zhou，Narendra Shenoy，William Nicholls
3. 《算法设计与分析》 王晓东